

DAVID STEWART

Sampling Rates for Analog Sensors

Why use trial-and-error methods to determine sampling rates when you can use science and mathematics? Here are the details of a simple procedure that makes more sense.

If you need to sample a push-button, opto-switch, resolver, pressure, chemical, or other sensor in real-time, how fast should you sample the input? Every 10ms? Every 30ms? How do you know what's the best rate? In an earlier article ("How to Choose a Sensible Sampling Rate," July 2002, p. 20), I related a conversation I had with one engineer. I asked what was the best sampling rate for his particular application.

His answer was "5ms."

"Why?" I asked.

"Because it works," he said. "We spent days testing a variety of sampling rates, and that's one that works."

In another application, the specifications listed a sampling rate that was formulated based solely on its use in a similar application. Including this value in the specifications left no flexibility to the system designer. What if that rate wasn't actually the best sampling rate for that application? If the new software appeared to work with the specified sampling rate, the rate would stay fixed for the lifetime of the application, with the value never questioned. The sampling rate would only be questioned if the sampling software didn't work.

In my previous article, I explained a way to scientifically determine the best sampling rate for digital inputs. In this article, I'll review that information and discuss how it applies to analog sensors.

It shouldn't take days of trial-and-error testing to determine the correct sampling rate when systematic engineering analysis of the application can yield correct answers with a few experiments.

Pick a number

Ad-hoc methods of choosing the sampling rate raise several issues.

First, it shouldn't take days of trial-and-error testing to determine the correct sampling rate for a given application. Rather, systematic engineering analysis of the application and problem at hand can yield correct answers with just a few experiments.

Second, what is the definition of "best"? The best answer for one application is not necessarily the best answer for another application. Often, trial-and-error methods are used to determine the sampling rate, and the testing is performed on a minimal system that includes only the sensor sampling code—not the entire application. Remember the engineer who answered "5ms" to my question? His code was created for a push-button switch that required some debouncing. Through testing, he ultimately selected a 5ms polling time because it didn't appear to erroneously register a single push as two and was sufficiently fast to not misinterpret an intentional double-push for a bounce.

In reality, a 5ms polling time might be acceptable for that system, but without consideration of other factors—in particular, the real-time response of the system—it's difficult to say whether it's the best answer.

For example, what if the processor is overloaded and the 5ms sampling is using 40% of the CPU's capacity? Increasing the sampling interval to 10ms would halve the CPU utilization. An alternative might be to execute the control code at half the speed. From a system perspective, which is better? Or more importantly, is there a good compromise between the resources used for sampling, and the effect that sampling has on processor

use and other factors like real-time schedulability and priority inversion?

When selecting a sampling rate, there are usually several competing goals, such as:

- Sample as fast as possible to obtain greatest accuracy.
- Sample as slow as possible to conserve processor time.
- Sample slow enough that noise doesn't dominate the input signal.
- Sample fast enough to provide adequate response time.
- Sample at a rate that's a multiple of the control algorithm frequency to minimize jitter.

The truth is there's usually no best answer for all systems, but there's often one answer that stands out as better than most others when the peculiarities of a specific application and the target hardware are considered.

In this article, I show how to systematically identify a set of good sampling rates through a combination of experimentation and mathematical analysis, and I discuss how to select the right value from that set given the real-time requirements of the application.

Here's a systematic approach to determining sampling rate:

1. Measure sensor characteristics of application.
2. If there is noise in the input, select the algorithm that will be used to filter the data.
3. Compute the lower and upper bound for sampling rates based on function alone.
4. Identify the trade-offs between using the lower and upper bound rates.
5. Prioritize the trade-offs, to determine a suitable sampling rate

between the computed lower and upper bounds.

This method combines experimental measurement with analytical understanding of the application's needs, in order to engineer a good solution. While the approach can be used for most types of sensors, in this article we focus only on analog inputs.

Analog inputs

Analog inputs provide data to the processor through an analog-to-digital converter (ADC). The sampling rate refers to the number of times the data is read from the ADC and passed along to other application components that use the data. The sampling rate directly affects the temporal resolution of the input signal, much in the same way as the number of bits of resolution in the ADC affects the spatial resolution.

The maximum error is a function of the sampling rate. We define the error $\epsilon(t)$ as the difference between the real sensor value and the value used by the control algorithm at any time, t . Note that t is continuous, thus as the sampling rate T_s increases, the input value is constant, the error typically increases. This is shown in Figure 1.

In signal processing, the Nyquist criterion is used to determine the sampling rate. Specifically, the Nyquist criterion states that the sampling rate must be at least twice as fast as the highest frequency component in the input signal. Given such a sampling rate, the original input signal can then be reconstructed.

Unfortunately, the Nyquist criterion cannot be used in most embedded control applications when reading analog sensors. Reconstruction of the original signal requires significant

For microcontrollers, as when using Nyquist criteria, the highest frequency component in the system can be used to determine the minimum sampling rate, f_{min} .

computational power; thus the need for digital signal processors. On the other hand, in embedded control systems, the analog input does not need to be reconstructed. Rather, the input is typically used to provide sensory input as the basis for feedback control. Thus, only the most recent data is needed. So, the question is how "recent" must that data be to keep the error within the maximum bounds specified by the application.

Finding a range of rates

For microcontrollers, as when using Nyquist criteria, the highest frequency component in the system can be used to determine the minimum sampling rate, f_{min} . Let's call that highest frequency component F . We define $\omega=2\pi F$. The worst-case changes in the analog input can be modelled as:

$$g(t) = 2^{n-1} \sin(\omega t) \quad (1)$$

where n is the number of bits on the ADC. The maximum rate of change for $g(t)$, which we call G , occurs when the derivative of $g(t)$ is maximized. Thus:

$$G = \frac{d}{dt} g(t) \Big|_{\max} = \omega 2^{n-1} \cdot \cos(\omega t) \Big|_{\omega t=0} = \omega 2^{n-1} \quad (2)$$

For many applications, the maximum rate of change might already be specified as a maximum slope, such as "one degree per second" for a digital thermometer. This can be converted to the form of $\Delta a / \Delta t$ (ADC units per second) through simple scaling based on the range of the ADC.

Other times, the maximum can be estimated reasonably through experimentation. For example, suppose an analog velocity sensor is connected to a motor carrying minimal load. The maximum rate of change of the velocity

occurs when the motor reaches steady state after full power. The measurement can use the analog sensor as input, by sampling as fast as the processor is capable of and recording the values of the velocity in ADC units.

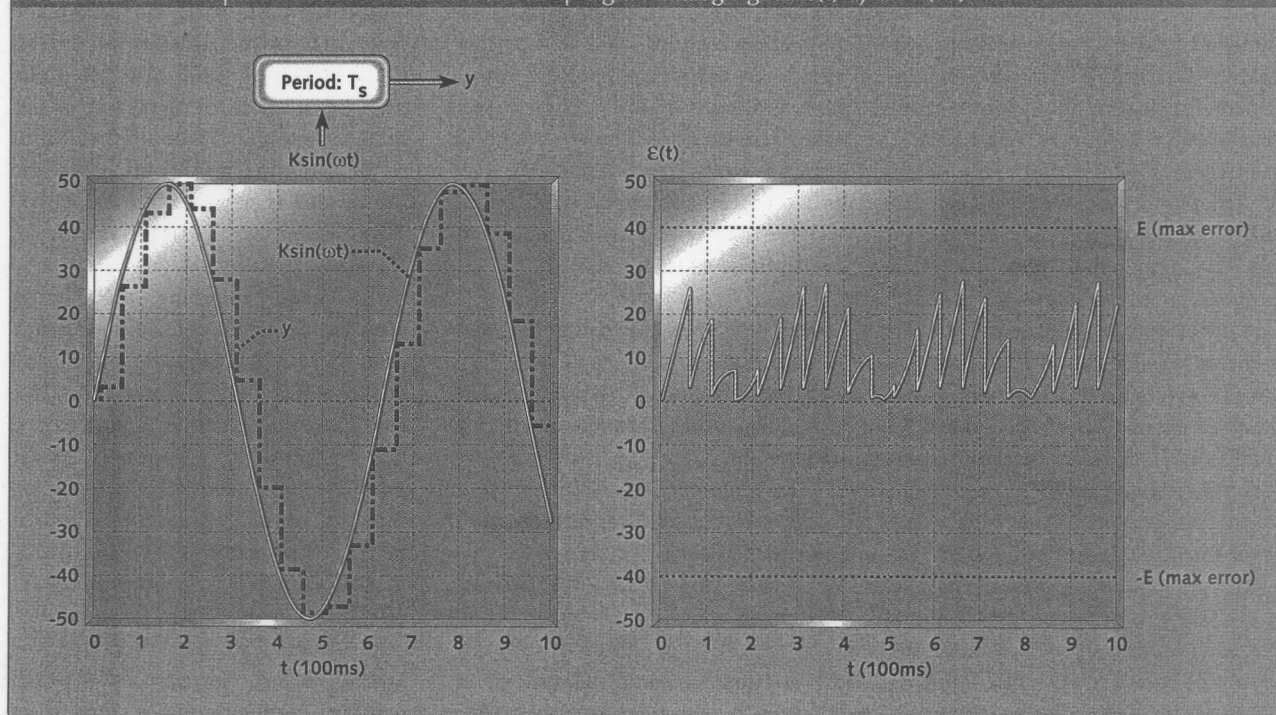
Suppose the sampling rate is Δt . Reading the log of measurements will yield a Δa during a time interval Δt . If Δa is the difference between two successive ADC readings and $\Delta a = \max(\Delta a)$, we can then compute $\Delta a / \Delta t$. If significant digits on the input Δa becomes an issue because Δt is too small, we can combine multiple samples and instead compute $k\Delta a / k\Delta t$, where k is the number of samples and $k > 1$.

Note that this Δt is for this experimentation phase only; the objective is to find T_s (the best value of Δt for the final application) to ensure sufficient accuracy while minimizing resource usage.

Since $\Delta a / \Delta t$ is a specification of the maximum slope, which by definition is also G , we note the relationship:

$$G = \frac{\Delta a}{\Delta t} = \omega 2^{n-1} = 2^n \pi F \quad (3)$$

FIGURE 1 Example of error that results from sampling an analog signal $\varepsilon(t) = |y - K \sin(\omega t)|$



For a specific ADC device and application, it doesn't make sense to sample any faster than at the rate f_{max} as the error is already limited by the resolution of the device.

Let E be the maximum allowable error, specified as a percentage of the maximum range of the signal. That is, for a maximum 5% error, $E=0.05$.

The minimum sampling rate for the ADC, f_{min} , is then computed as:

$$f_{min} = \frac{G}{E \cdot 2^n} = \frac{(\Delta A / \Delta t)}{E \cdot 2^n} = \frac{\omega}{2E} = \frac{\pi F}{E} \quad (4)$$

For example, if an application has a maximum rate of change that is equivalent to sampling a 50Hz sine wave, an 8-bit ADC is used to read a sensor, and the maximum error is 5%. Then, $f_{min} = 50\pi / 0.05 = 3.1\text{kHz}$. The sampling period $T_{min} = 1 / f_{min} = 318\mu\text{s}$.

An interesting note: the minimum sampling rate is not a function of the number of bits on the ADC. Rather, the number of bits creates a bound on the maximum error:

$$E \leq 2^{-n} \quad (5)$$

This leads to the question, if maximum error is specified as 1 ADC unit—which is the best resolution of the ADC—then what is the sampling rate that would yield this result? This corresponds to $\Delta A = 1$. To compute that value, we set $E = 2^{-n}$ in Equation 4, to provide:

$$f_{max} = G = 2^n \pi F \quad (6)$$

where f_{max} is the sampling rate to obtain maximum resolution.

For a specific ADC device and application, it doesn't make sense to sample any faster than at the rate f_{max} as the error is already limited by the resolution of the device.

The sampling rate for an analog input has a lower bound as defined in Equation 4 and an upper bound as defined in Equation 6. Thus to summarize, given F ,

$$\frac{\pi F}{E} \leq f_s \leq 2^n \pi F \quad (7)$$

or if given $\Delta A / \Delta t$:

$$\frac{(\Delta A / \Delta t)}{E \cdot 2^n} \leq f_s \leq \frac{\Delta A}{\Delta t} \quad (8)$$

Note that these equations provide a bound that assumes a maximum error based on the full range of the ADC, which is from 0 through 2^{n-1} . In some applications, the full range of the ADC might not be used, in which case the actual ADC range should be substituted for 2^n .

For example, to determine the valid range for f_s , consider a temperature sensor used in a controlled high-power heating system. The sensor connects to a 10-bit ADC with a 50 μs conversion time. The system is designed to operate between -20°C and 200°C . Furthermore, the control algorithm requires that the error in the sensor reading be no more than 0.5°C at any given point, to ensure stable control.

During the initial phases of development, an experiment is performed to determine $\Delta A / \Delta t$. The sensor is cooled to -20°C inside a small thermally insulated enclosure (for example, an oven). The heater is placed inside this enclosure and used to heat up the sensor and nothing more. Heating up more would result in a larger load on the system, and thus would slow the rate at which heating occurs. Code to sample the sensor and log the data is written and executed. The heater is then turned on to full power. Once the sensor is heated to 200°C , the heater is turned off. At this point, the maximum rate of cooling in an environment that reflects the real application is measured. If cooling is passive, then simply opening the door of the enclosed thermal enclosure might be sufficient. If cooling is active (for exam-

ple, a fan), then that device should be turned on.

The goal at this point is to obtain approximate measurements. A more elaborate setup, such as one that reflects the real load of the system, might be needed to fine-tune results. But these first estimates can provide a fairly good first-order approximation of the needed sampling rates to use as a guideline in the design and implementation of the system.

Suppose it took 380 seconds to heat the sensor from -20°C to 200°C . Looking more closely at the data, we found that over any 1s interval, the maximum temperature increase was 3.5°C . Assuming the 10-bit ADC is calibrated so that an ADC value of 0 corresponds to -20°C , a value of 0x3FF corresponds to 200°C , and the relationship is linear, 3.5°C per second corresponds to 16 ADC units per second. Therefore, $\Delta A / \Delta t = 16$. Also, since the maximum specified error is 0.5°C over the 220°C range, this translates to a maximum 0.23% error for the sensor reading. That is, $E = 0.0023$.

Using Equation 8, we compute the range of sampling rates for the analog sensor, which yields:

$$\frac{16}{0.0023 \cdot 2^{10}} \leq f_s \leq 16$$

or:

$$6.8\text{Hz} \leq f_s \leq 16\text{Hz}$$

This means that sampling slower than 6.8 times per second could cause an error that's outside the allowance dictated by the control system algorithm, especially during periods of maximum heating or cooling. The upper bound on the sampling frequency is 16 times per second. Note, however, that unlike digital inputs with bouncing, sampling analog sensors too fast won't cause errors. Rather, the upper limit on the frequency identifies the threshold at which faster sampling no longer improves the accuracy of the control system.

Finding the best rate

Now that you have a range of acceptable values, the next step is to consider various attributes of the application and hardware to determine the best sampling rate within the range of acceptable rates. Here are some issues you may encounter when looking for the best rate.

You may notice that the actual conversion time for the ADC didn't affect the sampling rate. This is often the case when the conversion time is negligible as compared to the sampling rate. On the other hand, suppose the conversion time were 60ms (this conversion time is more common with high-resolution ADCs). With such a long conversion time, it would be desirable to configure the ADC for continuous mode operation, and set the sampling rate to 16.67Hz, which is an exact multiple of the conversion rate. Sampling at half this rate, 8.33Hz, is also acceptable, if you want to save processor time. Since this rate is within the computed range, the application should still operate correctly. However, the analog input should not be sampled at quarter rate (4.16Hz), because this is outside the range, and would result in more error than allowed by the specifications. Operating at any frequency that isn't a multiple of the conversion time yields undesirable clock skew, and thus likely causes more problems, even if the sampling rate is within the computed range.

As another example of fine-tuning, suppose you determine that the control system needs more accurate sensor readings. This means the percent error in the sensor reading, E , must be reduced. This, in turn, results in an increase of the lower bound of f_{min} , meaning the sensor would need to be sampled more quickly. You can use Equation 8 to quickly determine the new minimum rate given the revised specification for E . Such calculations enable adjustments in sampling rate to be deter-

You'll often need to modify this approach depending on the particular sensors you're using, the needs of the application, and the ability to obtain reasonable measurements through simple experiments.

mined analytically within minutes, rather than taking hours or days trying to adjust it experimentally.

Note that this example considers using analog input data without filtering. If filtering of the input is needed to reduce noise, then the analysis must take into consideration the needs of the filtering algorithm, as shown in my previous article ("How to Choose a Sensible Sampling Rate," July 2002, p.20).

To summarize, the range provides a starting point for the design. Should any parameters change, you can adjust the computation accordingly and revise the sampling rate as necessary.

A sampler

This article presents an engineering approach towards determining good sampling rates for reading analog sensors that provide continuous data. Rather than using a single value for the sample rate, you can use experimental analysis to derive ranges based on application parameters. You can employ the equations in this article to quickly determine the minimum and maximum sampling rates (or periods) and adjust the sampling rates when fine-tuning the application. Based on other aspects of your application, such as processor use, the rate of the control or decision-making algorithm, and hardware constraints, you can then select the best value for the sampling rate from within the computed range.

While this article features analog input devices that are representative of those in many embedded systems, it's not a complete list of such devices. Rather than providing a solution that works for every possible analog sensor, the article demonstrates a combined analytical and experimental approach. You'll often need to modify this approach depending on

the particular sensors you're using, the needs of an application, and your ability to obtain reasonable measurements through simple experiments. The point to remember is that practicing fundamental engineering when developing software for embedded systems can save significant time in the development process and produce answers that are usually better and at least as good as any an-ad-hoc approach. **esp**

Dave Stewart is chief technology officer of Embedded Research Solutions. Dave was previously director of the Software Engineering for Real-Time Systems Laboratory at the University of Maryland. His primary area of expertise is in developing component-based software, tools, and frameworks for resource-constrained real-time embedded systems. He has a PhD in computer engineering from Carnegie Mellon University. His email address is dstewart@embedded-zone.com.

Further reading

- Dibble, Peter. "Deadline Scheduling," *Embedded Systems Programming*, March 2001, pp. 72-80.
- Kalinsky, David. "Context Switch," *Embedded Systems Programming*, February 2001, pp. 94-105.
- Smith, A. "The Merest Flick of a Switch," *Practical Electronics*, April 1991, pp. 24-29.
- Stewart, David B. "How to Choose a Sensible Sampling Rate," *Embedded Systems Programming*, July 2002, pp. 20-27.
- Stewart, David B. and P.K. Khosla. "Real-Time Scheduling of Sensor-Based Control Systems," in *Real-Time Programming*, Ed. W. Halang and K. Ramamritham, Pergamon Press, pp. 139-144, 199.

Acknowledgments

Research that led to the results described in this article was funded, in part, by National Science Foundation award #0000439.